CHARUSAT
CHAROTAR UNIVERSITY OF SCIENCE AND TECHNOLOGY

CE143: COMPUTER CONCEPTS & PROGRAMMING

Chapter – 8

Character Arrays

DEPSTAR

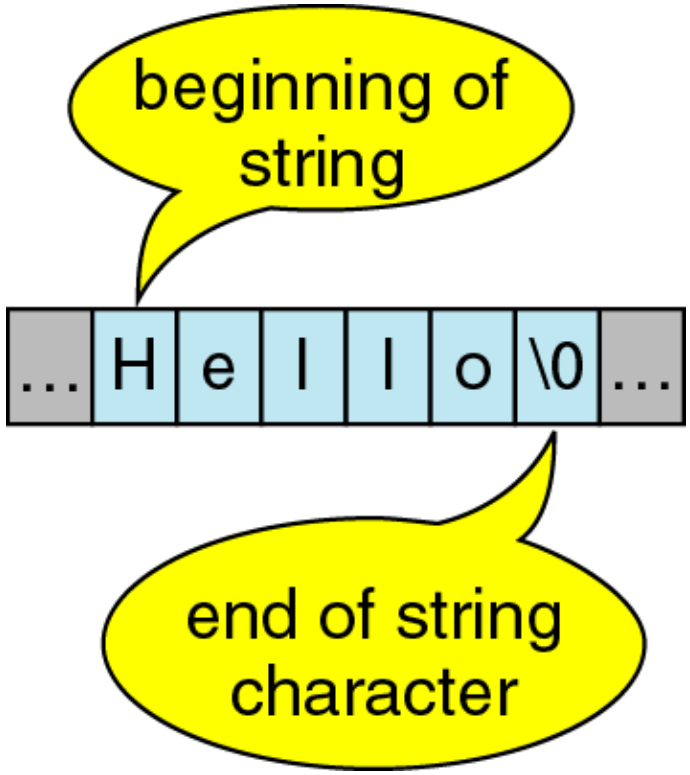
Devang Patel Institute of Advance Technology And Research

Objectives

- To understand character array and importance of NULL character.
- To understand Declaration, Initialization and various input and output methods of string, formatted output of string, arithmetic operations on characters.
- To understand various functions of string.h: strlen, strcat, strcmp, strcpy, strrev, strstr, etc.
- To learn Two dimensional character array (table of strings).

Introduction

- A sequence of characters is often referred to as a character “string”.
- A string is stored in an array of type **char** ending with the null character '\0 '.
- For example: The string "hello world" contains 12 characters including '\0' character which is automatically added by the compiler at the end of the string.



Introduction

- Any group of characters(except double quote sign) defined between double quotation marks is a string constant.
- Example:

  "Man is obviously made to think."

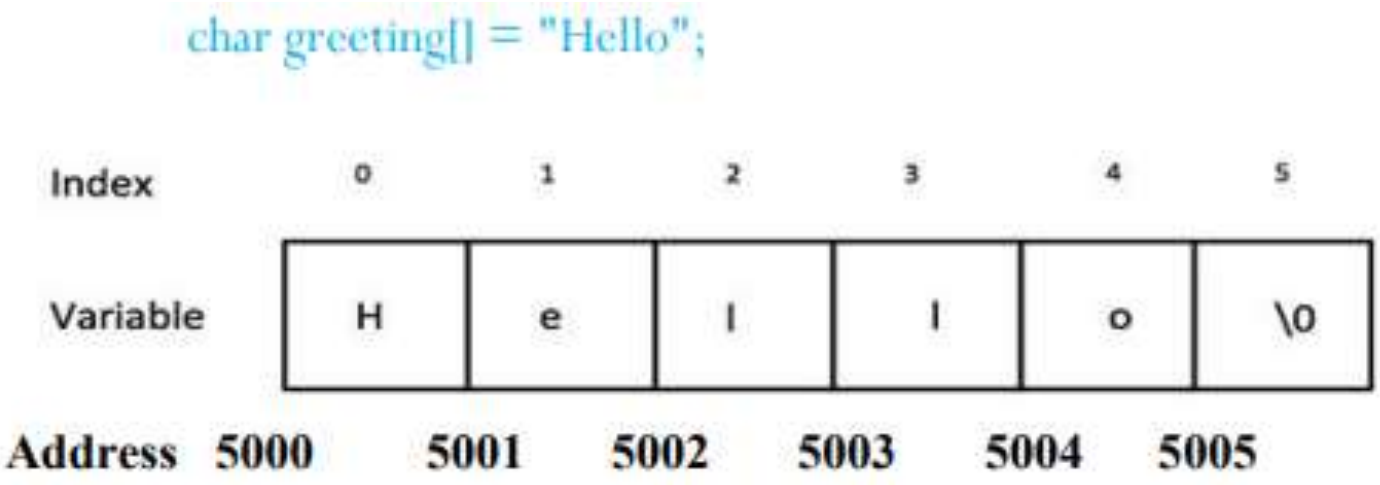
- To add double quotation in the string to be printed, back slash will be used.
- Example:

  printf("\" Well Done!\"");

- output: "Well Done!"

Importance of NULL character

- The terminating null ('\0') is important, because it is the only way the functions that work with a string can know where the string ends.
- In fact, a string not terminated by a '\0' is not really a string, but merely a collection of characters.
- Memory Representation of String:



Declaration & initializing string variables

- Character strings are often used to build meaningful and readable programs.
- The operations that are performed on character strings are
  - Reading and writing strings.
  - Combining strings together.
  - Copying one string to another.
  - Comparing strings for equality.
  - Extracting a portion of a string.

Declaration & initializing string variables

- C also permits to initialize a character array without specifying the number of elements.

```
char string[ ]={'G','O','O','D','\0'};
```

defines the array string as a five element array.

- We can also declare the size much larger than the string size in the initializer.

```
char str[10]="good";
```

- The following declarations are illegal:

```
char str2[3] = "good";
```

```
char str3[5];
str3="good“;
```

```
char s1[4] ='abc';
char s2[4];
s2=s1;  /*error*/
```

Reading strings from terminal

- The familiar input function scanf can be used with %s format specification to read in a string of characters.
- Example:

```
char address[15];
scanf("%s",address);
```

- scanf function terminates its input on the first white space it finds.

Reading a line of text

- It is not possible to use scanf function to read a line containing more than one word.
- This is because the scanf terminates reading as soon as a space is encountered in the input.
- We can use the getchar function repeatedly to read single character from the terminal, using the function getchar.
- Thus an entire line of text can be read and stored in an array.
- Syntax for getchar function is

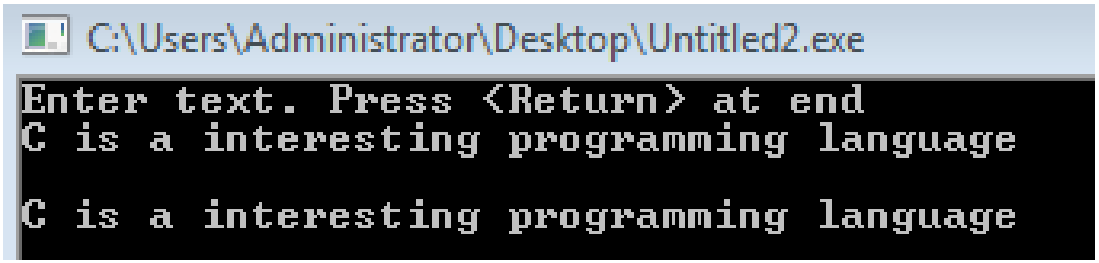
```
char ch;
ch = getchar();
```

- Note that the getchar function has no parameter.
- Another method of reading a String of text containing whitespace is to use gets function from <stdio.h> header file.

```
gets(str);
```

Use of getchar functions

```
#include  <stdio.h>
  void main( )
  {
     char  line[81], character;
     int   c;
     c = 0;
     printf("Enter text. Press <Return> at end\n");
     do
     {
        character = getchar();
        line[c]   = character;
        c++;
     }
     while(character != '\n');
     c = c - 1;
     line[c] = '\0';
     printf("\n%s\n", line);
  }
```



Use of gets functions

```
int main()
{
        char data[100];
        printf("Enter a String for gets() :");
        //get string input using gets(..) function
        gets(data);

        printf("Entered Data Is : will be with puts() :");
        //print string using puts(..) function
        puts(data);

   // Wait For Output Screen
   getch();
   //Main Function return Statement
   return 0;
}
```

Difference between scanf() and gets(), getchar()

- scanf() ends taking input upon encountering a whitespace, newline or EOF.
- gets() considers a whitespace as a part of the input string and ends the input upon encountering newline or EOF.
- getchar() is used to read a single character from a keyboard on the output screen. This character can be any character from the keyboard (a-z, A-Z, 0–9, !@#$%^&*()-=+_{}:"?><,./;'[]\|….any character)

Writing string to screen

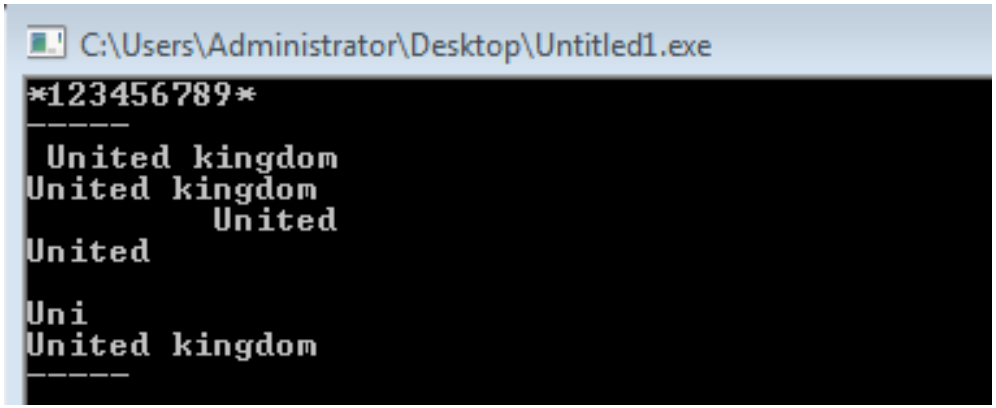
- We have used extensively the printf function with %s format to print strings to the screen.
- The format %s can be used to display an array of characters that is terminated by the null character.
- For example, the statement

  printf("%s", name);

  can be used to display the entire contents of the array name.

Writing string to screen-Example

```
void main()
{
char country[15] = "United kingdom";
printf("*123456789*\n");
printf("-----\n");
printf("%15s\n", country);
printf("%5s\n", country);
printf("%15.6s\n", country);
printf("%-15.6s\n", country);
printf("%15.0s\n", country);
printf("%.3s\n", country);
printf("%s\n", country);
printf("-----\n");
}
```



Writing string to screen-Example

```
void main()
{
int c,d;
char string[ ] = "CProgramming";
printf("-----\n");
for(c=0;c<=11;c++)
{
      d=c+1;
      printf("%-12.*s\n",d, string);
}
printf("-----\n");
for(c=11;c>=0;c--)
{
      d=c+1;
      printf("%-12.*s\n",d, string);
}
printf("-----\n");
}
```

Use of putchar and puts functions

- The function puts() is used to print strings while putchar() function is used to print character as their names specifies.
- These functions are from the stdio.h class doing the jobs related to strings.
- Example:

```
puts("I AM A STRING");
Output: I AM A STRING

putchar("a");
Output: a
```

putchar & puts functions-Example

```
int main ()
{
// initializing the variables
char a[15], b[15], c;

// coming string in a and b
strcpy(a, "includehelp");
strcpy(b, "ihelp");

// put a and b
puts(a);
puts(b);

// printing characters
for(c = 'Z' ; c >= 'A' ; c--)
{
        putchar(c);
}
return(0);
}
```

```
includehelp
ihelp
ZYXWVUTSRQPONMLKJIHGFEDCBA
Process returned 0 (0x0)   execution time : 0.029 s
Press any key to continue.
```

Arithmetic operations on characters

- C allows us to manipulate characters the same way we do with numbers.
- Whenever a character constant or character variable is used in an expression, it is automatically converted into integer value by the system.
- For eg, if the machine uses the ASCII representation, then,

```
x = 'a';
printf("%d \n",x);
```

will display the number 97 on the screen.

Arithmetic operations on characters

- **Way 1: Displays ASCII value[Note that %d in Printf]**

```
char x = 'a';
printf("%d",x); // Display Result = 97
```

- **Way 2 : Displays Character value[Note that %c in Printf]**

```
char x = 'a';
printf("%c",x); // Display Result = a
```

- **Way 3 : Displays Next ASCII value[Note that %d in Printf]**

```
char x = 'a' + 1 ;
printf("%d",x);
// Display Result = 98 ( ascii of 'b' )
```

Arithmetic operations on characters

- **Way 4: Displays Next Character value[Note that %c in Printf]**

```
char x = 'a' + 1;
printf("%c",x); // Display Result = 'b‘
```

- **Way 5 : Displays Difference between 2 ASCII in Integer[Note %d in Printf]**

```
char x = 'z' - 'a';
printf("%d",x);
// Display Result = 25 (difference between ASCII of z and a )
```

- **Way 6 : Displays Difference between 2 ASCII in Char [Note that %c in Printf]**

```
char x = 'z' - 'a';
printf("%c",x);
// Display Result = ( difference between ASCII of z and a )
```

String Function

- atoi = A to I = Alphabet to Integer
- Convert string of number into integer
- Syntax:

```
num = atoi(String);
num   - Integer Variable
String- String of Numbers
```

- Example :

```
num = atoi("1947");
printf("%d",num);
```

- Output :

```
1947
```

- Significance :
  - Can Convert any string of number into Integer Value that can perform the arithmetic operations like integer
  - Header file : stdlib.h

Ways of using atoi Function

- **Way 1 : Passing Variable in atoi Function**

```
// Variable marks is of char Type
    int num;
    char marks[3] = "98";
    num = atoi(marks);
    printf("\nMarks : %d",num);
```

- **Way 2 : Passing Direct String in atoi Function**

```
    int num;
    num = atoi("98");
    printf("\nMarks : %d",num);
```

Putting strings togather

- Just as we cannot assign one string to another directly, we cannot join two strings together by the simple arithmetic addition.
- That is, the statements such as

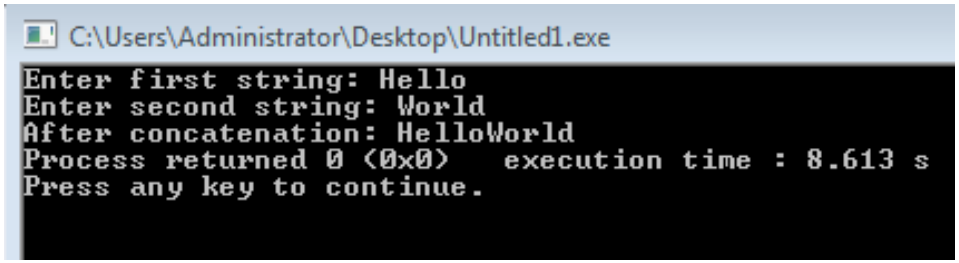
  ```
  string3 = string1 + string2;
  string2 = string1 + “hello”;
  ```

  are not valid.
- The characters from string1 and string2 should be copied into string3 one after the other.
- The process of combining two strings together is called concatenation.

Putting strings togather

Concatenate Two Strings Without Using strcat()

```
#include <stdio.h>
int main()
{
   char s1[100], s2[100], i, j;
   printf("Enter first string: ");
   scanf("%s", s1);
   printf("Enter second string: ");
   scanf("%s", s2);
   // calculate the length of string s1 and store it in i
   for(i = 0; s1[i] != '\0'; ++i);
  {
            for(j = 0; s2[j] != '\0'; ++j, ++i)
               {
                  s1[i] = s2[j];
               }
             s1[i] = '\0';
   printf("After concatenation: %s", s1);
   return 0;
}
}
```



Comparision of two strings

- C does not permit the comparison of two strings directly.
- That is, the statements such as are not permitted.

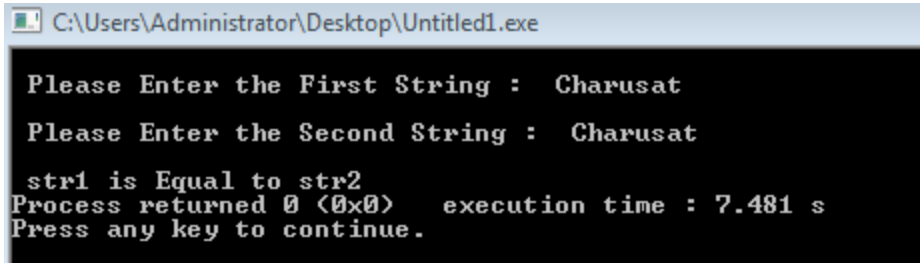
```
if(name1 == name2)
if(name == “ABC”);
```

- It is therefore necessary to compare the two strings to be tested, character by character.
- The comparison is done until there is a mismatch or one of the strings terminate into a null character, whichever occurs first.

Comparison of two strings

```
int main()
{
        char Str1[100], Str2[100];
        int result, i;
        printf("\n Please Enter the First String :  ");
        gets(Str1);
        printf("\n Please Enter the Second String :  ");
        gets(Str2);

        for(i = 0; Str1[i] == Str2[i] && Str1[i] == '\0'; i++);
        {
         if(Str1[i] < Str2[i])
                printf("\n str1 is Less than str2");
        else if(Str1[i] > Str2[i])
                printf("\n str2 is Less than str1");
        else
                printf("\n str1 is Equal to str2");
        }
       return 0;
}
```



String handling functions

- C library supports a large number of string-handling functions that can be used to carry out many of the string manipulation activities.
- Following are the most commonly used functions:
  - **strlen() function :** Finds the length of the string
  - **strcat() function :** Concatenates two strings
  - **strcmp() function :** Compares two strings
  - **Strcpy() function :** Copies one string over another
  - **Strncpy() function :** Copies portion of contents of one string into another string.
  - **strstr() function :** Finds first occurrence of sub-string in other string

strlen( ) Function

- strlen( ) function in C gives the length of the given string.
- Syntax for strlen( ) function is given below.

  n=strlen(string);
- strlen( ) function counts the number of characters in a given string and returns the integer value.
- It stops counting the character when null character is found. Because, null character indicates the end of the string in C.

strlen( ) Function

```
#include <stdio.h>
#include <string.h>

int main( )
{
   int len;
   char array[50]="CHARUSAT UNIVERSITY" ;

   len = strlen(array) ;

   printf ( "\string length  = %d \n" , len ) ;
   return 0;
}
```

strcat( ) Function

- The strcat function joins(concatenates) two strings together.
- Syntax is:
  - strcat (str2, str1); – str1 is concatenated at the end of str2.
  - strcat (str1, str2); – str2 is concatenated at the end of str1.
- As you know, each string in C is ended up with null character ('\0').
- In strcat( ) operation, null character of destination string is overwritten by source string's first character and null character is added at the end of new destination string which is created after strcat( ) operation.

```
strcat(strcat(string1,string2),string3);
```

- Here three strings are concatenated and the result is stored in string1.

strcat( ) Function- Example

- In this program, two strings “Welcome to” and “CHARUSAT” are concatenated using strcat( ) function and result is displayed as “Welcome to CHARUSAT”.

```
void main( )
{
  char source[ ] = " CHARUSAT" ;
  char target[ ]= "Welcome to" ;
  printf ( "\nSource string = %s", source ) ;
  printf ( "\nTarget string = %s", target ) ;
  strcat ( target, source ) ;
  printf ( "\nTarget string after strcat( ) = %s", target ) ;
}
```

strcmp( ) Function

- strcmp( ) function in C compares two given strings and returns zero if they are same.
- If length of string1 < string2, it returns < 0 value.
- If length of string1 > string2, it returns > 0 value.
- Syntax for strcmp( ) function is given below.

```
int strcmp (str1, str2 );
```

- strcmp( ) function is case sensitive. i.e, “A” and “a” are treated as different characters.

strcmp( ) Function - Example

```
#include <stdio.h>
#include <string.h>
int main( )
{
   char str1[ ] = "fresh" ;
   char str2[ ] = "refresh" ;
   int i, j, k ;
   i = strcmp ( str1, "fresh" ) ;
   j = strcmp ( str1, str2 ) ;
   k = strcmp ( str1, "f" ) ;
   printf ( "\n%d %d %d", i, j, k ) ;
   return 0;
}
```

strcpy( ) Function

- strcpy( ) function copies contents of one string into another string.
- Syntax for strcpy function is given below.

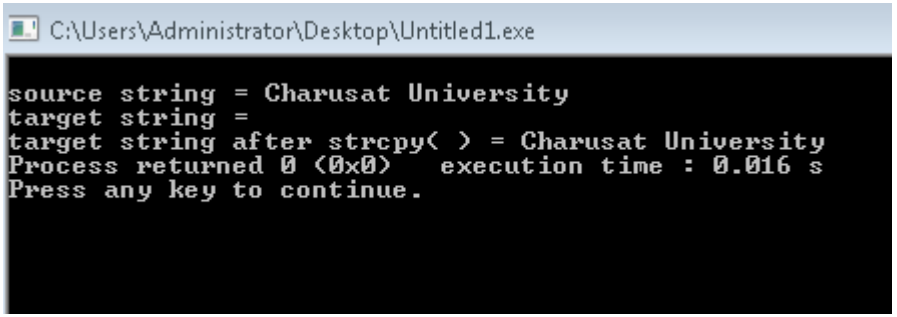
  strcpy (str1, str2); – It copies contents of str2 into str1.

  strcpy (str2, str1); – It copies contents of str1 into str2.
- If destination string length is less than source string, entire source string value won't be copied into destination string.
- For example, consider destination string length is 20 and source string length is 30. Then, only 20 characters from source string will be copied into destination string and remaining 10 characters won't be copied and will be truncated.

strcpy( ) Function

```
#include <stdio.h>
#include <string.h>

int main( )
{
   char source[ ] = "Charusat University" ;
   char target[20]= "" ;
   printf ( "\nsource string = %s", source ) ;
   printf ( "\ntarget string = %s", target ) ;
   strcpy ( target, source ) ;
   printf ( "\ntarget string after strcpy( ) = %s", target ) ;
   return 0;
}
```



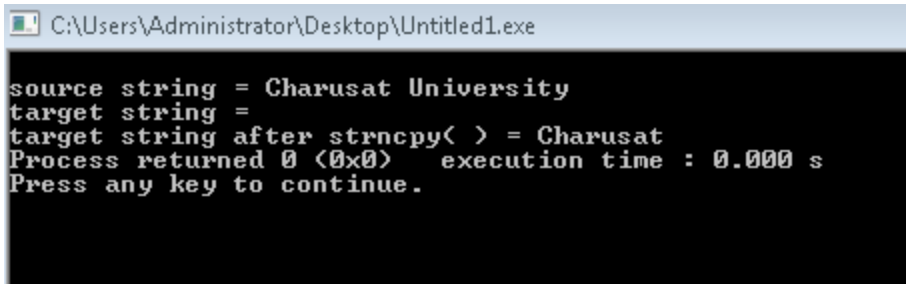
strncpy( ) Function

- strncpy( ) function copies portion of contents of one string into another string.
- Syntax for strncpy( ) function is given below.
  strncpy (str1, str2, 4); – It copies first 4 characters of str2 into str1.
  strncpy (str2, str1, 4); – It copies first 4 characters of str1 into str2.
- If destination string length is less than source string, entire source string value won't be copied into destination string.
- For example, consider destination string length is 20 and source string length is 30.
- If you want to copy 25 characters from source string using strncpy( ) function, only 20 characters from source string will be copied into destination string and remaining 5 characters won't be copied and will be truncated.

strncpy( ) Function

```
#include <stdio.h>
#include <string.h>

int main( )
{
   char source[ ] = "Charusat University" ;
   char target[20]= "" ;
   printf ( "\nsource string = %s", source ) ;
   printf ( "\ntarget string = %s", target ) ;
   strncpy ( target, source, 8 ) ;
   printf ( "\ntarget string after strncpy( ) = %s", target ) ;
   return 0;
}
```



strstr( ) Function

- It is two parameter function that can be used to locate a sub-string in a string.
- This takes the form:

```
strstr(s1,s2);
strstr(s1,"ABC");
```

- The function strstr() searches the string s1 to see whether the string s2 is contained in s1.
- If yes, the function returns the position of the first occurance of the sub-string. Otherwise, it returns a NULL pointer.

```
if (strstr(s1,s2)== NULL)
        printf("substring is not found");
else
        printf("s2 is substring of s1");
```

Other string functions

String functions	Description
strcat ( )	Concatenates str2 at the end of str1
strncat ( )	Appends a portion of string to another
strcpy ( )	Copies str2 into str1
strncpy ( )	Copies given number of characters of one string to another
strlen ( )	Gives the length of str1
strcmp ( )	Returns 0 if str1 is same as str2. Returns <0 if strl < str2. Returns >0 if str1 > str2
strcmpi ( )	Same as strcmp() function. But, this function negotiates case. "A" and "a" are treated as same.
strchr ( )	Returns pointer to first occurrence of char in str1
strrchr ( )	last occurrence of given character in a string is found
strstr ( )	Returns pointer to first occurrence of str2 in str1
strdup ( )	Duplicates the string
strlwr ( )	Converts string to lowercase
strupr ( )	Converts string to uppercase
strrev ( )	Reverses the given string
strset ( )	Sets all character in a string to given character
strnset ( )	It sets the portion of characters in a string to given character
strtok ( )	Tokenizing given string using delimiter

table of string & other features

- A list of names can be treated as a table of strings and a two-dimensional character array can be used to store the entire list.

- For example, A character array **student[30][15]** may be used to store a list of 30 names each of length not more than 15 characters.

Table of string

- A list of names can be treated as a table of strings and a two-dimensional character array can be used to store the entire list.

C	H	A	N	D	I	G	A	D	G
M	A	D	R	A	S				
H	Y	D	R	A	B	A	D		
A	H	M	E	D	A	B	A	D	
M	U	M	B	A	I				

```
char city[ ][ ]
{
    "CHANDIGADH"
    "MADRAS"
    "HYDRABAD"
    "AHMEDABAD"
    "MUMBAI"
}
```

- To access the name of the ith city, we write city[i+1].

Previous Year Questions

- Differentiate gets and scanf functions?
- Explain any four string handling functions in detail.
- Explain strcmp() and strcpy() with example.
- Write a program to copy a given string into another string without using string.h file.
- Explain strlen(), strcmp(), strcpy() functions?
- Write a program to copy the last n characters of a character array in another character array. Also convert the lower case letters into upper case letters while copying.